Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 25 (2016) 582 – 589

Procedia Technology

Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016)

An Implementation of Modified Lightweight Advanced Encryption Standard in FPGA

Mary James[a] , Deepa S Kumar [b]*

[a]*PG Scholar,College Of Engineering Munnar,Munnar,Idukki, Pin-685612,India*
[b]*HOD,College Of Engineering Munnar,Munnar,Idukki, Pin-685612,India*

Abstract

Advanced Encryption Standard (AES) is the standardized block cipher, which is used in various applications. AES is well suited for software and hardware implementation with versions of 128,192,256 key sizes. In hardware implementation AES is advantageous as it is more secure, low cost, and has minimized hardware utilization. Lightweight block ciphers are developed for the efficient implementation in hardware. An approach to design a technique to implement AES as lightweight block cipher is an immediate requirement of the time. An approach, to make AES a lightweight block cipher, is being discussed such that designing the steps of AES such as mix columns, substitute byte in AES is to be implemented in a parallel manner. The latency in this implementation is considered to be less comparing the conventional implementation of AES. The conventional and the new approach are to be simulated in XILINX 14.2 and is being compared in the aspects of area and latency. The design is to be implemented in FPGA.

Peer-review under responsibility of the organizing committee of RAEREST 2016

Keywords: AES; Lightweight cryptography;parallel mixcolumns; Parallel Substitute bytes; FPGA

1. Introduction

AES algorithm was developed in the year 2000 with high security [1]. AES is a substitution and permutation network in which the algorithm has only substitution using S-Box and the permutations involve shifting and other

* Mary James. Tel.: +91 9048782766.
E-mail address: maryjames1408@gmail.com

Peer-review under responsibility of the organizing committee of RAEREST 2016
doi:10.1016/j.protcy.2016.08.148

mathematical alterations. AES is the standardized block cipher which is being used in many applications nowadays. The need of the time is to develop a block cipher to implement it as lightweight. The aim of this proposal is to improve throughput of the AES algorithm. The algorithm implemented in conventional method(Rijndael) and the improved version of AES algorithm with parallel mixcolumns and parallel substitution bytes are to be analyzed. The key generation is to be done in each step instead of generating it once and storing it, as the storage takes much space in implementation. The substitution bytes can be again optimized by using look up tables for the S-box implementation. The techniques are to be analyzed in terms of area, power and latency.

The proposal focuses on the improvement of the throughput of AES by decreasing latency. The architectures will be developed and analysis will be done using Xilinx ISE design suite 14.2.The latency of these architectures will be analyzed. The results will be compared with conventional available Rijndael algorithm.

- Improving the throughput of the encryption algorithm.
- Implementation of this design will result in lightweight encryption algorithm and reduced latency such that it can be efficiently implemented in FPGA.

In order to increase the throughput of the encryption parallelism is used in the blocks which cause more delay of the system. Since Mix Columns block has more delay parallelism is applied to that block. This will reduce the delay, thereby increasing the throughput. The Parallel Mix Columns is used in order to reduce the latency. In this the block computes one column at a time such that the four columns are executed at the same time rather than each byte executing at a time. In Parallel Sub Bytes, four columns are executed at the same time rather than each byte executing at a time, this reduces the latency.

2. Literature Survey

AES is an iterative and symmetric block cipher with the steps such as mixcoumns, shiftrows, substitute bytes, Add round key for the encryption and the inverse of these steps for decryption. The AES Rijndael is a symmetric block cipher developed by Joan Daemen and Vincent Rijmen in the year 2000 which was selected by NIST [1]. Considering the properties such as security , efficiency, performance , flexibility, implementation ability the Rijndael algorithm has the best overall scores considering other algorithms.The block size is 128 bits and key sizes can be 128,192 and 256 bits. Based on the key sizes the number of iterations can be 10,12 or 14. The bock cipher consists of 4 steps. The block is divided into 4x4 bytes of matrix. The steps involved are Substitution Bytes , ShiftRows, Mixcolumns and AddRound Key [2]. This step involves substitution of bytes using S-box. The next step is shift rows step, this involves no change in first row, a single shift in second row , shift twice in third row and shift thrice in forth row. The next step is mixcoloumn step in which all four bytes in the column replace each byte of a column. In the next step , the key is XORed bitwise with the state known as AddRound key. The inverse of these steps are done in decryption process to obtain the plain text [3]. The advantages of this cipher is that it has resistance against all known attacks, in wide range of platforms it provides speed and code compactness, and the design simplicity is also a major advantage.

It is very important to optimize the parameters such as area, latency and power. The main aim is to develop AES into lightweight block cipher. There are many implementations that deal with the optimizations. One such implementation is having pipelined combinational S-Box [6]. This implementation requires huge area as it has repeated use of S-Box. This increases latency and no efficiency is improved in this method. Another technique is to use look up table based S-Box design [7]. This design results in low complexity and also reduces latency. Thus improving throughput of the algorithm. Another implementation consists of BRAM based S-Box design [8]. The precomputed values are stored in the BRAM. This has input wires, so the design consumes high area. In another implementation of AES instead of Galois field multiplication, shift and add multiplication is used. But this implementation has greater delay compared to other implementation. Considering FPGA implementation, the most time taking process is round key generation. This wastes nearly 12 clock cycles of the FPGA. So in order to reduce this latency, another implementation has the generation of key for the round in each round itself instead of generating it once and storing it.

The conventional method of implementing AES has more latency in steps such as mixcolumns and substitution bytes. In order to reduce these latency the steps can be implemented in parallel manner [10]. The use of S-Box and

other mathematical steps are done in a parallel manner. The implementation of AES should be therefore in a lightweight manner. There is a comparison of some of the lightweight block ciphers which were developed after the establishment of AES. These block ciphers are substitution and permutation networks (SPN) which are modifications of AES.A comparison of some of them are given in the following table.

Table 1. Survey on lightweight SPN block ciphers

Name	structure	block size	Key size	Number of rounds
AES	SPN	128	128,198,256	10,12,14
Klein	SPN	64	64,80,96	12,16,20
LED	SPN	64	64,128	32,48
Mysterion	SPN	128,256		12,16
Zorro	SPN	128	128	24
mCrypton	SPN	64	64,96,128	12
Fantomas/Robin	SPN	128	128	12
Noekeon	SPN	128	128	16
Present	SPN	64	128	31
Pride	SPN	64	128	20
Prince	SPN	64	128	10
Rectangle	SPN	64	80,128	25

From the above table it is clear that AES is having lesser number of rounds with higher bits of block and key sizes.

3. The Proposed method

Lightweight block ciphers are the block ciphers that are applied in sensor nodes and RFID tags. Lightweight block ciphers are designed to achieve a low cost implementation and an efficient hardware design. AES128 is to be designed in a resource constraint manner with 10 rounds to be implemented in hardware. The different steps are implemented in efficient manner to achieve resource constraint in latency. Thus increasing throughput of the design. The AES encryption is an iterative algorithm and uses four operations in different rounds, namely SubBytes, ShiftRows, MixColumns and Key Additions transformations. The initialization is done by adding first round key (128 bits) with 128 bits plain text. In subsequent steps, the following transformations are done: SubBytes, ShiftRows, MixColumns and AddRoundKey [3]. The last round is different from the previous rounds as there is no MixColumns transformation. The decryption process involves of the inverse steps, decryption round contains of: Inverse S-BOX used for Byte Substitution, Inverse Shift Rows, Add Round Key and Inverse MixColumns. The round keys will be generated in each round instead of generating it once.

3.1 Subbytes/inverse subbytes transformation

The first transformation, SubBytes, is used for encryption and inverse SubBytes used for decryption. The SubBytes substitution is a nonlinear byte substitution that operates independently on each byte of the State using a substitution table (S-box). Take the multiplicative inverse in the finite field Galois field and affine transform to do the SubBytes transformation. Inverse affine transform have to find for inverse SubBytes transformation then multiplicative inverse of that byte. Figure indicates that how the transformation can be done. The step is done in parallel in this methodology. The corresponding value of the row and column are replaced from the values corresponding to it in S-Box. Similarly the inverse is done in inverse substitution of bytes [4]. The Galois field multiplication is a tedious process. Therefore in the design the values are already calculated and are being substituted whenever necessary. This is done using look up table. Subbytes is processed in a parallel manner in order to reduce latency.

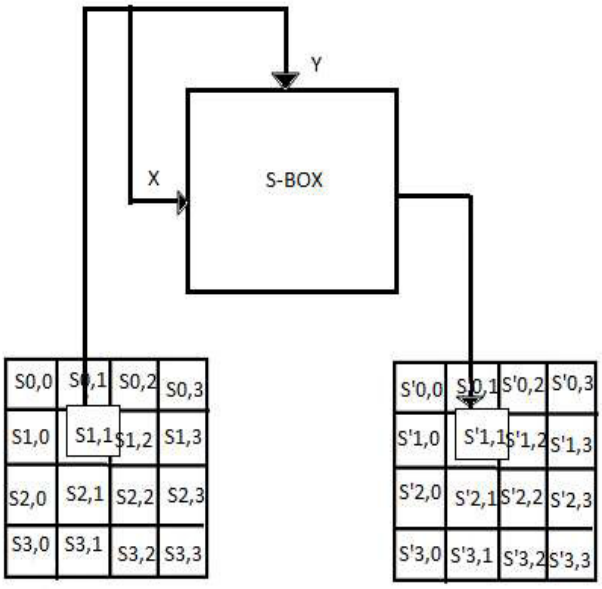

Fig.1 The illustration for Substitute bytes transformation.

3.2 Shift rows/inverse shiftrows transformation

The transformation is called ShiftRows performs in encryption, in which rows are cyclic shifting to the left. The number of shifting depends upon the row number of the state matrix [4]. First row is not shifted in the design whereas the second row is shifted by one byte,third row is shifted to two bytes and fourth row is shifted to three byte left. Similarly in the decryption, InvShiftRows transformation performs the right cyclic shifting operation inverse of ShiftRows; number of shifting depends on number of row number. First row is not shifted. Second row shifted to right once, third row shifted twice and forth row shifted thrice to right.

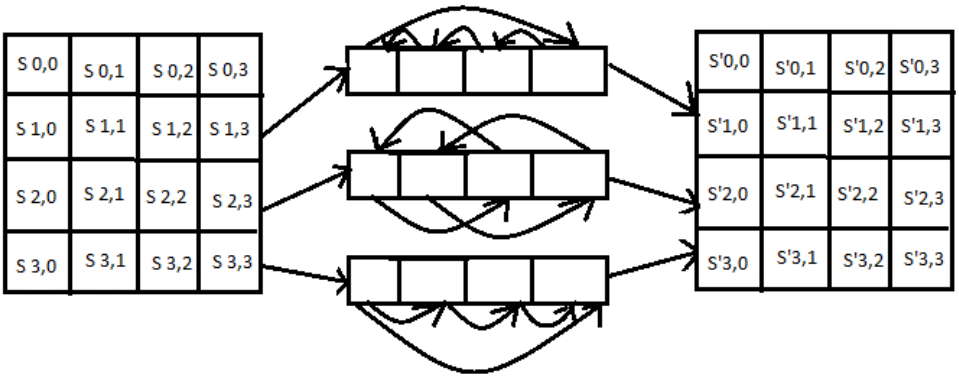

Fig 2 The illustration of Shift rows transformation.

3.3 Mixcolumns/inverse mixcolumns tranformation

Mixcolumns operate on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column. The transformation is defined by the matrix multiplication on state. This transformation is based on Galois Field multiplication. Each byte of a column is replaced with another value that is a function of all four bytes in the given column. The Mix Columns transformation operates on the State column by column, treating each column as a four term polynomial.

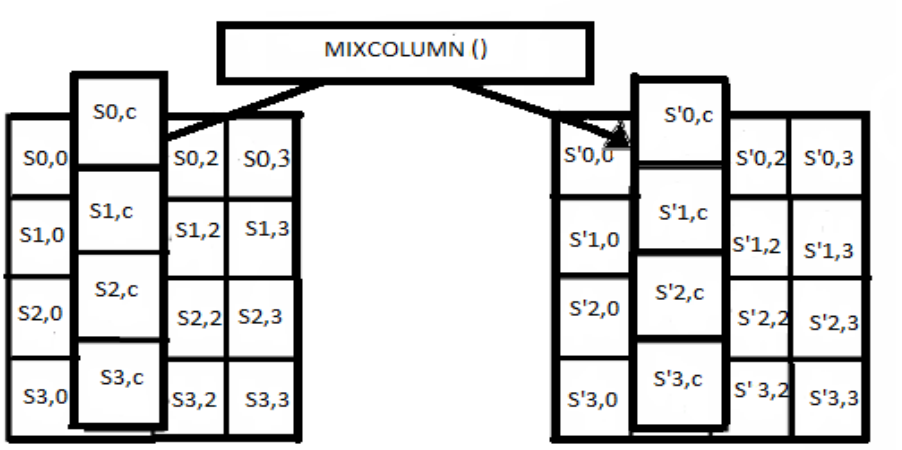

Fig 3 The mixcolumns transformation of AES.

The mixcolumn step takes much time to process each column. Therefore the step of mixcolumn is done in a parallel manner in order to reduce the latency.

3.4 Add round key

The Add Round Key operation is a simple XOR operation between the State and the Round Key. The Round Key is derived from the Cipher key by means of the key schedule. The State and Round Key are of the same size (128 bits) and to obtain the next State an XOR operation is done each element.

3.5 Key generation of AES

Key generation in AES is also termed as key expansion which takes a 4-word (16 byte) key and produces a linear array of 44 words(156 bytes). Usually the key generation is done once as the initial step and is stored. The round keys are taken whenever necessary. But in this design we aim to generate the round keys in each round. This can reduce the area required to store the round keys.

3.6 Encryption and decrytion Process

AES encryption goes through 10 rounds of encryption process. In the first 9 rounds the input data is mapped into Sub Bytes, Shift Rows and Mix Columns, and in the 10th round the Mix Columns operation is not included then Add Round Keys is performed. Similarly in decryption the 10 rounds of the steps are done and in the last round the inverse mix column is not performed.

The above described structure of AES with the rounds in iterative manner will result in a low latency and high throughput design.AES is symmetric and iterative block cipher. The approach will result in design which will result in lightweight version of AES to be implemented in FPGA.

4. Work done

In this work the proposed design coding is done in VHDL, and simulation and synthesis was done in Xilinx ISE Design Suite 14.2. Different steps of AES encryption are designed. These designs are synthesized using Xilinx ISE Design Suite 14.2. The different transformation for the encryption such as Substitute bytes, Shiftrows, Add round key, Mixcolumns are designed in VHDL language. The inverse of shift rows and substitute bytes are also implemented for the decryption process.

4.1 Simulation Results of steps of AES

4.1.1 Subbytes/inverse subbytes transformation

The corresponding row and column of the state matrix is substituted with the value from the S-Box. Similarly the inverse subbytes takes the value from inverse S-Box.

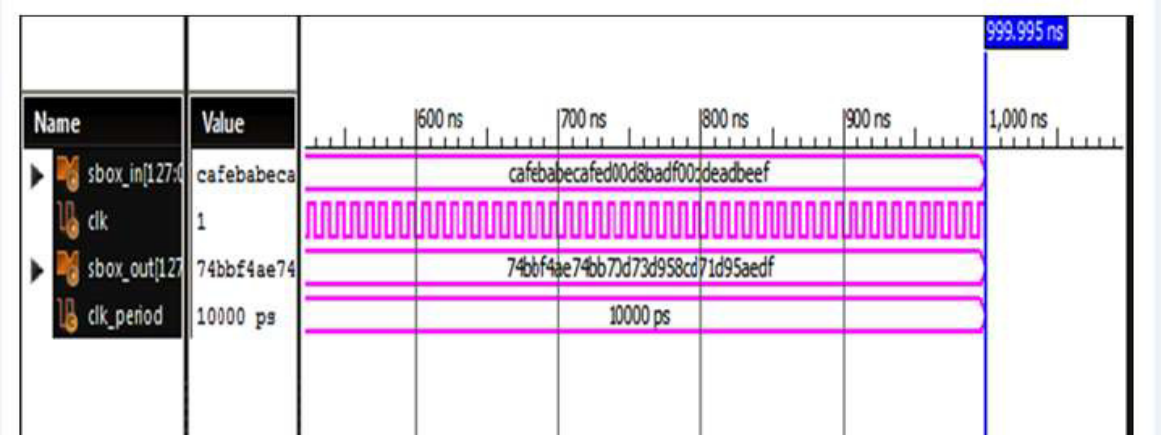

Fig 4 Simulation result of Subbytes transformation

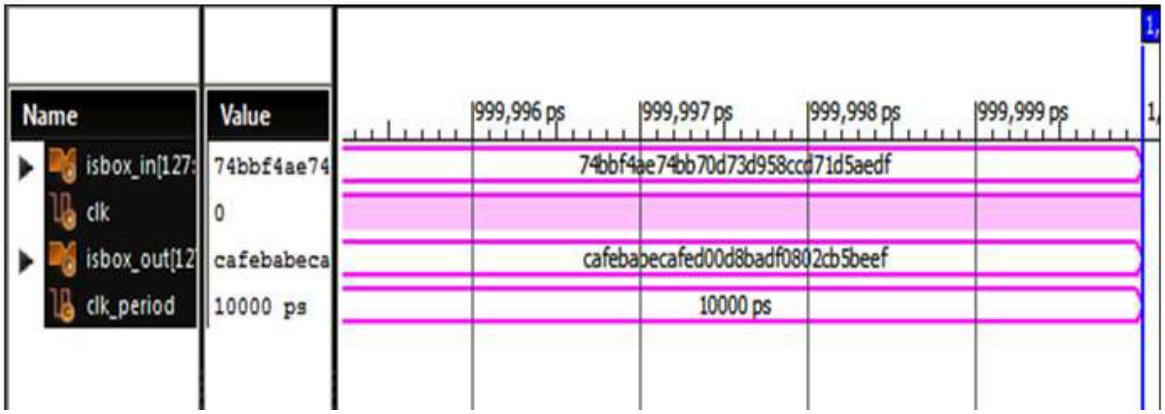

Fig 5 Simulation result of inverse Subbytes transformation

4.1.2 Shift rows/inverse shiftrows transformation

Each row in the state matrix of AES is being shifted as explained in the previously. The simulation results are shown as below.

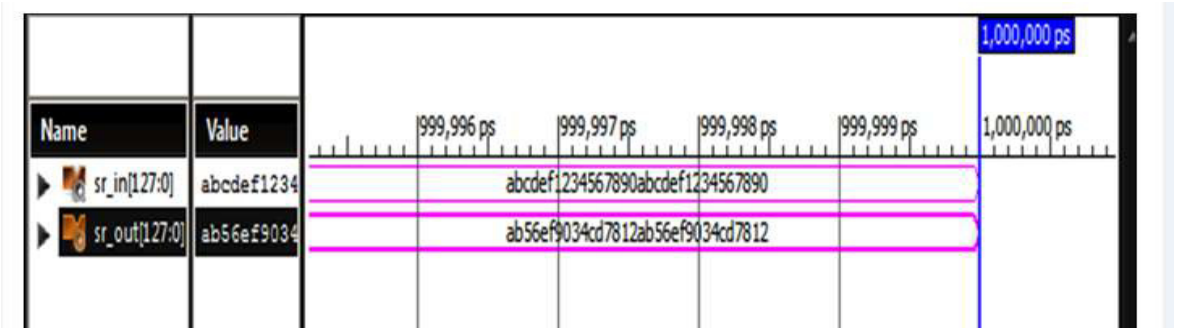

Fig 6 Simulation result of shiftrows transformation

Fig 7 Simulation result of inverse shiftrows transformation

4.1.3 Mixcolumns transformation

The columns of the state matrix is being altered as was explained in the previous session. The simulation result of the mixcolumns transformation is being shown as follows.

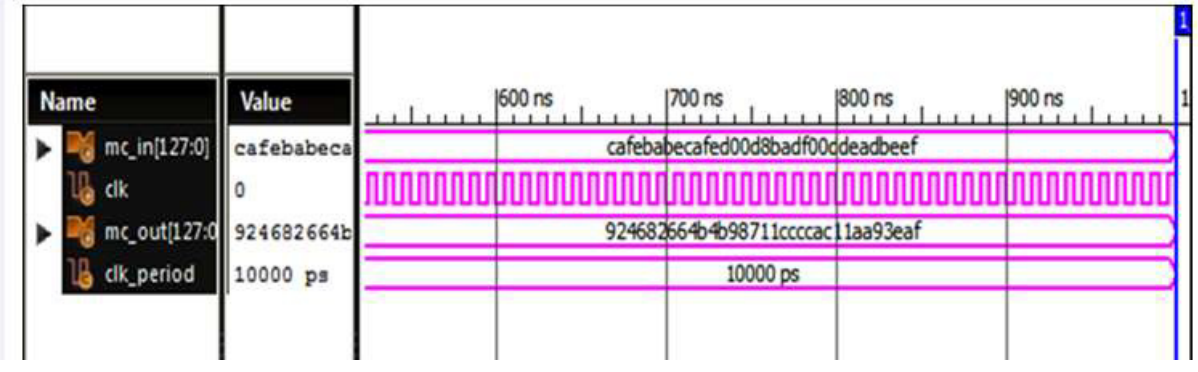

Fig 8 Simulation result of mixcolumns transformation

4.1.4 Add round key transformation

The elements of the state matrix is being XORed with the key of 128 bits in add round key step.The simulation result of the Add round key transformation is being shown as follows.

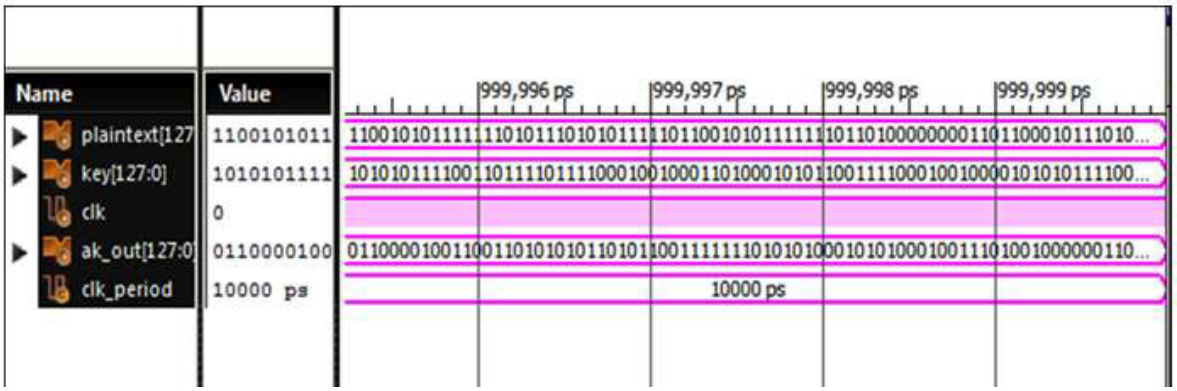

Fig. 9 Simulation result of Add round key transformation

5. Conclusion

Lightweight block ciphers are designed to achieve a low cost implementation and an efficient hardware design. AES 128 is to be implemented in XILINX 14.2 in an efficient manner by conventional model and few steps such as mix columns and substitution bytes are to be applied in a parallel manner. The key generation is done once in conventional method. This wastes lots of clock cycles in FPGA . Therefore a method is used in which the key generation is done in each round. The latency is expected to be reduced in this implementation. AES is one of the substitution and permutation based block cipher. This design is to be implemented in XILINX 14.2 and is to be compared in terms of latency. The simulation of different steps of encryption and decryption are done in XILINX 14.2.Spartan 6 is the FPGA that is to be used to implement AES.

6. References

[1] J. Daemen and V. Rijmen, “The rijndael block cipher: Aes proposal,” in First Candidate Conference (AES1), pp. 343–348, 1999.
[2] S. William and W. Stallings, Cryptography and Network Security, 4/E. Pearson Education India, 2006.
[3] A. M. Deshpande, M. S. Deshpande, and D. N. Kayatanavar, “Fpga implementation of aes encryption and decryption,” in Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009 International Conference on, pp. 1–6, IEEE, 2009.
[4] P. B. Ghewari, J. Patil, and A. B. Chougule, “Efficient hardware design and implementation of aes cryptosystem,” International journal of engineering science and technology,vol. 2, no. 3, pp. 213–219, 2010.
[5] A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, “An fpga-based performance evaluation of the aes block cipher candidate algorithm finalists,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 9, no. 4, pp. 545–557, 2001.
[6] O. S. Gomes, R. L. Moreno, and T. C. Pimenta, “A fast cryptography pipelined hardware developed in fpga with vhdl,” in Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on, pp. 1–6, IEEE, 2011.
[7] T. Hoang et al., “An efficient fpga implementation of the advanced encryption standard algorithm,” in Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on, pp. 1–4, IEEE, 2012.
[8] P. N. Khose and V. G. Raut, “Implementation of aes algorithm on fpga for low area consumption,” in Pervasive Computing (ICPC), 2015 International Conference on, pp. 1–4, IEEE, 2015.
[9] J. Tay, M. Wong, and I. Hijazin, “Compact and low power aes block cipher using lightweight key expansion mechanism and optimal number of s-boxes,” in Intelligent Signal Processing and Communication Systems (ISPACS), 2014 International Symposium on, pp. 108–114, IEEE, 2014.
[10] O. S. Dhede and S. Shah, “A review: Hardware implementation of aes using minimal resources on fpga,” in Pervasive Computing (ICPC), 2015 International Conference on, pp. 1–3, IEEE, 2015.